***Stack Overflow***

*Moldovan Raul Alexandru*

*Group: 30432*

Table of Contents

[1. Introduction 3](#_Toc163000078)

[2. Tech stack 3](#_Toc163000079)

[3. SW architecture 3](#_Toc163000080)

[3.1. Model layer 3](#_Toc163000081)

[3.2. Repository layer 4](#_Toc163000082)

[3.3. Service layer 4](#_Toc163000083)

[3.4. Controller layer 4](#_Toc163000084)

[4. Functional requirements 5](#_Toc163000085)

[4.1. Feature 1 5](#_Toc163000086)

[4.2. Feature 2 5](#_Toc163000087)

[4.3. Feature 3 5](#_Toc163000088)

[5. Non-functional requirements 5](#_Toc163000089)

[6. Diagrams 6](#_Toc163000090)

[6.1. Class diagram 6](#_Toc163000091)

[6.2. Use case diagram 7](#_Toc163000092)

[6.3. Database diagram 7](#_Toc163000093)

[7. Testing 7](#_Toc163000094)

# Introduction

The simple Stack Overflow project aims to create a streamlined and user-friendly platform for asking and answering questions within a community. Drawing inspiration from the popular Stack Overflow website, our system provides a simplified version focusing on core features such as asking questions, providing answers, and voting on content.

# Tech stack

The Simple Stack Overflow project utilizes a modern and robust tech stack to achieve its objectives. The key technologies and frameworks employed in the development process include:

* **Spring Boot**: The project is built on the Spring Boot framework, providing a powerful and streamlined development environment for building enterprise-grade applications.
* **Spring Boot Starter Data JPA**: This dependency enables seamless integration with the Java Persistence API (JPA) for data access and manipulation.
* **Spring Boot Starter Web**: With the Spring Boot Starter Web dependency, the project leverages Spring's web capabilities to develop RESTful APIs and web applications.
* **MySQL Connector/J**: The MySQL Connector/J dependency enables connectivity to the MySQL database management system, allowing the application to interact with MySQL databases seamlessly.
* **Maven**: The project utilizes Apache Maven as a build automation tool and dependency management tool.
* **MySQL Workbench**: MySQL Workbench is employed as a visual database design and modeling tool.
* **IntelliJ IDEA**: IntelliJ IDEA serves as the integrated development environment (IDE) for the project.

# SW architecture

The Simple Stack Overflow project follows a typical MVC (Model-View-Controller) architecture pattern, where the application's components are organized into separate layers responsible for handling different concerns.

## Model layer

The model layer encapsulates the business logic and data representation of the application. It consists of the following entities:

* **User**: Represents user information such as username, password, and user score.
* **Question**: Represents a question posted by a user, including details such as title, text, creation date, tags, and associated answers.
* **Answer**: Represents an answer provided by a user for a specific question, containing the answer text, creation date, and associated question.

## Repository layer

The repository layer is responsible for interacting with the database and performing CRUD (Create, Read, Update, Delete) operations on the application's entities. It consists of repository interfaces that extend Spring Data JPA's JpaRepository:

* **UserRepository**: Handles database operations related to user entities.
* **QuestionRepository**: Manages database operations for question entities.
* **AnswerRepository**: Handles database operations for answer entities.

## Service layer

The service layer contains the business logic of the application and acts as an intermediary between the controllers and the repositories. It encapsulates the application's logic for processing user requests and coordinating data access. The service layer includes the following components:

* **UserService**: Provides services related to user management, such as user registration, authentication, and score computation.
* **QuestionService**: Manages operations related to questions, including posting questions, editing, deleting, and retrieving questions by tag or user.
* **AnswerService**: Handles operations related to answers, such as posting answers, editing, deleting, and retrieving answers for a specific question.

## Controller layer

The controller layer serves as the entry point for handling incoming HTTP requests and generating HTTP responses. It contains the RESTful API endpoints that clients interact with to perform various actions within the application. The controller layer includes the following classes:

* **UserController**: Handles user-related HTTP requests, such as user registration, authentication, and profile management.
* **QuestionController**: Manages HTTP requests related to questions, including posting questions, retrieving questions, and filtering questions by tag or user.
* **AnswerController**: Handles HTTP requests related to answers, such as posting answers, retrieving answers for a specific question, and managing answer voting.

# Functional requirements

## Feature 1

* Users shall be able to ask questions. Each question must have an author, title, text, creation date & time, picture and one or more tags. If an appropriate tag does not exist, the user must be able to create one.
* The list of questions shall be displayed, sorted by creation date. The most recent question should be displayed first.
* Questions may be edited or deleted by their author.
* The user must be able to filter questions by tag, via a text search, via users or for his own questions. The text search should check the question title.

## Feature 2

* Each question may be answered one or more times by any user (including the original author).
* Each answer must have an author, text, picture and creation date & time.
* Answers may be edited or deleted by their author.
* When displaying a question individually, the list of answers must also be displayed.

## Feature 3

* Users may vote questions and answers (upvote and downvote, like and dislike).
* Each user may only vote once on each question or answer. Users cannot vote on their own answers or questions (Like & Dislike).
* On each voted question or answer, the vote count must be displayed (vote count = upvote/like count - downvote/dislike count). The vote count can be negative.
* The answers for a question must be sorted by their vote count. Answers with the highest vote count must be displayed first.

# Non-functional requirements

* **Responsiveness**: System response time should be under 1 second for user interactions.
* **Data Protection**: Encrypt user data (passwords).
* **Authentication and Authorization**: Secure authentication and authorization mechanisms.
* **Intuitive UI**: User-friendly interface for easy navigation.
* **Modularity**: Well-structured, modular system.
* **Code Quality**: Adherence to coding standards and best practices.
* **Documentation**: Comprehensive documentation for architecture and codebase.

# Diagrams

## Class diagram

A screenshot of a computer

Description automatically generated

## Use case diagram

A diagram of a user

Description automatically generated

## Database diagram

A screenshot of a computer

Description automatically generated

# Testing

Tests were done using Postman.